

Local rewiring algorithms to increase clustering and grow a small world

Jeff Alstott,^{*} Christine Klymko,[†] Pamela B. Pyzza,[‡] Mary Radcliffe[§]

August 11, 2016

Abstract

Many real-world networks have high clustering among vertices: vertices that share neighbors are often also directly connected to each other. A network's clustering can be a useful indicator of its connectedness and community structure. Algorithms for generating networks with high clustering have been developed, but typically rely on adding or removing edges and nodes, sometimes from a completely empty network. Here, we introduce algorithms that create a highly clustered network by starting with an existing network and rearranging edges, without adding or removing them; these algorithms can preserve other network properties even as the clustering increases. These algorithms rely on local rewiring rules, in which a single edge changes one of its vertices in a way that is guaranteed to increase clustering. This greedy algorithm can be applied iteratively to transform a random network into a form with much higher clustering. Additionally, these algorithms grow the network's clustering faster than they increase its path length, meaning that network enters a regime of comparatively high clustering and low path length: a small world. These algorithms may be a basis for how real-world networks rearrange themselves organically to achieve or maintain high clustering and small-world structure.

1 Introduction

Many complex systems are organized as networks, including social networks, gene interaction networks, the world wide web, and beyond [6, 7]. The majority of complex networks from real-world systems have statistical properties that separate them from the random graphs studied more classically. These properties include a skewed or power-law degree distribution [2, 9], a high clustering coefficient [4, 19], and a low path length, creating a small world of connectivity [24].

A network's clustering coefficient is a measure of how often two connected nodes will have neighbors in common. More formally, the clustering coefficient relates a networks' number of triangles (3-cycles) to the possible number of triangles (given by the total number of wedges in the network). There are two common ways to define a clustering coefficient, which are related but different. The first, often called the global clustering coefficient, involves a global count of the number of triangles and possible triangles in a network. The second involves an average of local counts. Formal definitions can be found in Section 2. Real-world networks tend to have (relatively) high clustering coefficients under both definitions due to the presence of communities: if two nodes are in the same community, it not only increases the likelihood that there is an edge between them,

^{*}Massachusetts Institute of Technology/Singapore University of Technology and Design, alstott@mit.edu

[†]Lawrence Livermore National Laboratory, klymko1@llnl.gov

[‡]Ohio Wesleyan University, pbpyzza@owu.edu

[§]Carnegie Mellon University, mradclif@math.cmu.edu

but also that they have a common neighbor. Due to this, the clustering coefficient is a good measure of how well-connected a network is and whether or not there is any strong community structure present [15, 22].

Closely related to the concept of the clustering coefficient of a network is the idea of the “small world” property found in many real world networks. Informally, the small world property states that the average distance between any two nodes in the networks is relatively small and, in the case of an evolving network, the distance grows more slowly than the network grows. In networks that are not fully connected, having a short average distance is opposed to having a high clustering coefficient. This is due to the fact that in order for nodes in distant parts of the network to have short paths between each other, there must exist some long range edges, between communities. However, adding long range edges increases the number of possible triangles in a network without creating any new triangles, since nodes in different communities are unlikely to have neighbors in common, thus lowering the clustering coefficient. There has been research in how to jointly maximize these two properties under various constraints [3, 27] as well as to understand processes which may lead to the dual development of both properties in both real world and generated networks [8, 11, 16, 20, 21].

Most research on generating networks with high clustering or low path length has focused on building these networks from scratch. However, in many real systems a network is already in existence. One may want to minimally rearrange existing edges in order to enhance or reduce certain network properties. The goal of these rewirings could include be increasing network robustness [5, 12, 13, 14, 18, 25, 26], communicability [1], synchronizability [17] or algebraic connectivity [23].

Here we present edge rewiring algorithms which increases the clustering coefficient of a given network while minimally impacting other network properties, specifically degree distribution and average path length. We provide proofs that these algorithms monotonically increase the global clustering coefficient. We present basic notation and definitions in Section 2. We then describe the algorithms in Section 3, along with theoretical results about the effect of these algorithms on clustering. We numerically simulate the algorithms on model networks in Section 4. We discuss the implications of these algorithms in Section 5.

2 Notation and definitions

Let $G = (V, E)$ be a graph with a set of vertices (also referred to as nodes) V , $|V| = n$, and edge set $E = \{u \sim v \mid u, v \in V \text{ are connected by an edge}\}$, and write \bar{E} for the complement of E in $\binom{V}{2}$. For a vertex $v \in V(G)$, write d_v to denote the degree of vertex v . We define the *neighborhood* of v in G to be $N(v) = \{u \in V(G) \mid u \sim v\}$. To simplify notation, we shall denote by $\bar{N}(v)$ the complement of $N(v)$ in $V(G) \setminus \{v\}$; that is, $\bar{N}(v)$ is the set of vertices in G other than v itself to which v is not adjacent. Given two vertices x, y , define their *common neighborhood* to be $N(x, y) = \{u \in V(G) \mid u \sim x \text{ and } u \sim y\}$. For a set $S \subset V$, define $e(S)$ to be the number of edges in G having both endpoints in S .

Define $N_p(G)$ and $N_t(G)$ to be the number of length-two paths and number of triangles in the graph, respectively. Note that

$$N_p(G) = \sum_{x, y \in V(G)} |N(x, y)|,$$

and

$$3N_t(G) = \sum_{\{x, y\} \in E(G)} |N(x, y)|.$$

We define the clustering coefficient of G to be $C(G) = 3N_t(G)/N_p(G)$.

This version of the clustering coefficient is sometimes referred to as the global clustering coefficient, and we shall use that language at times in this work. Another measure of clustering, known as the local average clustering coefficient, is also in common usage. For a vertex $v \in V(G)$, this is defined as

$$c_v = \frac{2e(N(v))}{d_v(d_v - 1)}.$$

Here, we have that $\binom{d_v}{2}$ is the number of length-two paths having v at the center; that is, it is a measure of the number of triangles in which v could be involved. Then c_v can be seen as the proportion of these triangles that actually exist in the graph, compared to how many triangles are possible involving v . We then define the local average clustering coefficient to be $c(G) = \frac{1}{n} \sum_{v \in V} c_v$.

Although the global clustering coefficient and the local average clustering coefficient often produce similar results regarding clustering in the graph, they are not equivalent measures [10, p. 83]. The primary difference between the two measures is the emphasis placed on lower degree vertices. In the local average clustering coefficient, each vertex is treated with equal weight, and hence vertices of very low degree, for which c_v might be unusually high or low, have a much more substantial impact on the constant than those of high degree. In the global version, this impact is avoided by considering the graph as a whole, rather than its specific local structures.

The *adjacency matrix* associated with a graph G is given by $A = a(u, v)$ with

$$a(u, v) = \begin{cases} 1, & \text{if } u \sim v, \\ 0, & \text{else.} \end{cases}$$

Given two sets $A, B \subset X$, we use the notation $A \triangle B$ to denote the symmetric difference between A and B ; that is,

$$A \triangle B = (A \setminus B) \cup (B \setminus A).$$

3 Algorithms

In this section, we fully describe the algorithms used to rewire a graph to increase its clustering coefficient, and prove that these algorithms are monotonic with respect to the global clustering coefficient. Further, we examine some of the local optima of this algorithm and consider the impact of the rewiring procedure on the local average clustering coefficient.

3.1 Algorithm Description

To begin, let us examine the algorithm in question. We shall consider two different, but similar algorithms. Both of these are based on an iterated protocol that chooses one edge to rewire at each time step. The algorithms below describe how to complete one edge rewire; we then iterate to rewire as many edges as needed. The first version of the protocol is detailed in Rewiring Algorithm 1 below. Essentially, the protocol is as follows. We first choose a nonedge $\{x, y\}$ whose addition to the graph G would increase the number of triangles in G as much as possible. We then look among the existing incident edges for an edge whose removal would decrease the number of triangles in G as little as possible. If certain degree considerations are met, we then rewire this edge to $\{x, y\}$ to form a new graph G' . This is illustrated in Figure 1.

“Find Best” Rewiring Algorithm 1

- 1: Find a pair $\{x, y\} \in \bar{E}$ with the maximum value of $|N(x, y)|$
 - 2: For $v \in N(x)$, define $f_v = |N(x, v)|$, and for $v \in N(y)$, define $f_v = |N(y, v)|$. Choose the vertex $v \in N(x) \Delta N(y)$ with minimum f_v ; given a tie, choose v to have the highest possible degree. WLOG, suppose $v \sim x$.
 - 3: If $|N(x, v)| \geq |N(x, y)|$, return to step 1, and eliminate the edge $\{x, y\}$ from consideration.
 - 4: If $d_v > d_y$, rewire the edge vx to the edge yx to form G' .
 - 5: If $d_v \leq d_y$, return to step 2, and eliminate the vertex v from consideration.
 - 6: If no neighbor in $N(x) \Delta N(y)$ satisfies the requirements, return to step 1 and choose a different pair of nonadjacent vertices.
-

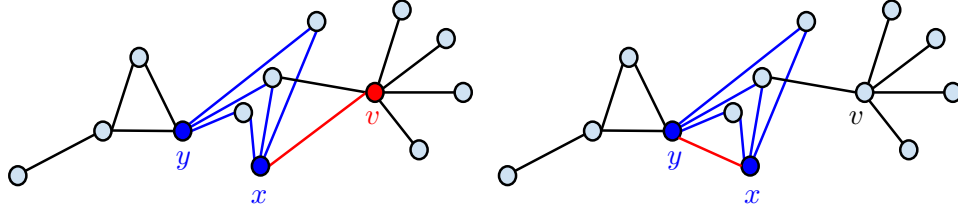


Figure 1: A graph G being rewired using Rewiring Algorithm 1. Here, the two blue vertices are x and y . Note that in the original graph (left), x and y have the largest number of common neighbors among nonedges of the graph. We then choose v as the red vertex, as its rewiring would destroy the fewest number of triangles, and it has maximum degree among such vertices. After verifying that all the degree considerations are met, we rewire the edge xv to form the new graph (right).

For the second version of the protocol, described below in Rewiring Algorithm 2, we take a slightly different approach to choosing an edge to rewire. In the first version, we first seek a nonedge whose addition would increase the number of triangles in G as much as possible. For the second version, we take the opposite approach, and choose an edge whose removal would destroy the fewest number of triangles in G as possible. This approach can be much faster when the graphs in question are more sparse; the remainder of the procedure is essentially the same as the first version of the algorithm.

“Improve Worst” Rewiring Algorithm 2

- 1: Find a pair $\{x, v\} \in E$ with the minimum value of $|N(x, v)|$
 - 2: For $y \in \bar{N}(x)$, define $f_y = |N(x, y)|$, and for $y \in \bar{N}(v)$, define $f_y = |N(y, v)|$. Choose a vertex $y \in \bar{N}(x) \Delta \bar{N}(v)$ with minimum f_y ; given a tie, choose y to have the lowest possible degree. WLOG, suppose $y \sim x$.
 - 3: If $|N(x, y)| < |N(x, v)|$, return to step 1, and eliminate the edge $\{x, v\}$ from consideration.
 - 4: If $d_y < d_v$, rewire the edge vx to the edge yx to form G' .
 - 5: If $d_y \geq d_v$, return to step 2, and eliminate the vertex y from consideration.
 - 6: If no neighbor in $\bar{N}(x) \Delta \bar{N}(v)$ satisfies the requirements, return to step 1 and choose a different pair of nonadjacent vertices.
-

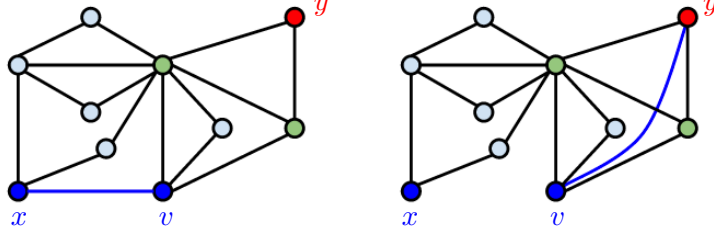


Figure 2: A graph G being rewired using Rewiring Algorithm 2. Here, the two blue vertices are x and v . Note that in the original graph (left), x and v have the fewest number of common neighbors among edges of the graph; namely, x and v are not involved in any common triangles. We then choose y as the red vertex, noting that rewiring xv to vy will add two triangles at the green vertices. After verifying that all the degree considerations are met, we rewire the edge to vy to form the new graph (right).

Theorem 1. *The algorithms described above strictly increase the clustering coefficient.*

Proof. We need only show that any legal rewiring will increase the clustering coefficient. To that effect, suppose we have three vertices x, y, v such that

- $\{x, y\} \in \bar{E}$, $\{x, v\} \in E$
- $|N(x, y)| > |N(x, v)|$
- $d_v > d_y$.

Then the rewiring of edge xv to edge xy is permitted according to either algorithm described above. We note that this situation accommodates both versions of the protocol, although the vertex labels are changed for the second version. Let $G' = G \setminus \{xv\} \cup \{xy\}$. Let us consider how this rewiring affects the clustering coefficient. Specifically, we need only consider the total number of triangles and the total number of length 2 paths in the new graph G' .

First, we consider triangles. We note that the only triangles that will be present in G but not in G' are those that have xv as an edge. On the other hand, the only triangles that will be present in G' but not in G are those that have xy as an edge. Hence, we have $N_t(G') = N_t(G) - |N(x, v)| + |N(x, y)| > N_t(G)$.

Likewise, the only length-two paths that appear in G but not G' are those involving the edge xv ; we note that there are $(d_v - 1) + (d_x - 1)$ such paths. Similarly, the only length-two paths that appear in G' but not in G are those involving the edge xy , of which there are $(d_x - 1) + d_y$. Hence, we have $N_p(G') = N_p(G) - (d_v - 1) - (d_x - 1) + (d_x - 1) + d_y = N_p(G) + d_y - d_v + 1 \leq N_p(G)$.

Therefore, we have that $C(G') = 3N_t(G')/N_p(G') > 3N_t(G)/N_p(G) = C(G)$. □

3.2 Local optima

We now turn to a consideration of local optima under this algorithm. As the algorithm is strictly monotone with respect to clustering, one would expect that any such optima will have a high clustering coefficient. Indeed, as the theorem below shows, this will be the case for our technique.

Theorem 2. *Let G be a graph, such that the edges of G can be partitioned into cliques of size at least 3, say C_1, C_2, \dots, C_k . Then G is a local optimum with respect to the above algorithm.*

Proof. We need only show that there are no legal rewires to be performed under the algorithm. We recall that in order to rewire vx to yx , we must have that $|N(x, v)| < |N(x, y)|$, and hence if $|N(x, y)| = 0$, there are no edges that can be rewired to xy . Moreover, note that any two cliques can share at most one vertex, as the cliques partition the edges of G .

Now, suppose that $x, y \in \bar{E}$. Now, if $d_x = 0$ or $d_y = 0$, then $|N(x, y)| = 0$ and there will be no legal rewires.

If not, both x and y appear as members of at least one clique. Note that they are not in the same clique, as if they were, we would have $x \sim y$. If the cliques including x and the cliques including y share no vertices, then $|N(x, y)| = 0$, and there will be no edge to rewire to xy .

Hence, we may assume that $x \in C_i$, $y \in C_j$, where $|V(C_i) \cap V(C_j)| = 1$. Thus, x and y share as a neighbor the vertex at which the two cliques intersect, and hence $|N(x, y)| = 1$. Now, note that if v is a neighbor of x , then v and x appear in some clique of size at least 3 together, and hence $|N(x, v)| \geq 1$. But then xv cannot be rewired to xy . As the same will be true of neighbors of y , there is no edge that satisfies the requirements of the algorithm.

Therefore, G is locally optimal with respect to the algorithm. \square

We note that the clustering coefficient of these graphs will be quite close to 1. Taking G to be as described in Theorem 2, define H to be the graph on $[k]$, wherein $i \sim j$ if and only if C_i and C_j share a vertex. We then have

$$\begin{aligned} C(G) &= \frac{3N_t(G)}{N_p(G)} \\ &= \frac{3 \sum_{i=1}^k \binom{n_i}{3}}{\sum_{i=1}^k n_i \binom{n_i - 1}{2} + \sum_{i \sim_H j} n_i n_j} \\ &\geq \frac{\sum_{i=1}^k n_i \binom{n_i - 1}{2}}{\sum_{i=1}^k n_i \binom{n_i - 1}{2} + \sum_{i,j} n_i n_j} \end{aligned}$$

Clearly, the fewer common vertices we have among cliques, the higher the clustering coefficient will be. Moreover, if we imagine that k is fixed, but the number of vertices in the graph is tending to ∞ , then $C(G)$ tends to 1 asymptotically.

To the best of these authors' knowledge, the optimum connected graph on n vertices with a fixed number of edges m with respect to the global clustering coefficient $C(G)$ is unknown. We note that these algorithms as written do not necessarily require that the edge rewiring preserves connectivity or components in the graph G , although it is clear based on the structure that this algorithm cannot combine two components into one. However, it is straightforward to construct examples in which one wishes to rewire a bridge, thus disconnecting a formerly connected graph (one such example is shown in Figure 3 below). Moreover, we note that there are local optima that do not take the form described above; for example, a barbell graph in which two complete graphs are connected by exactly one edge cannot be partitioned into cliques of size at least 3, but it is still optimal with respect to this rewiring algorithm.

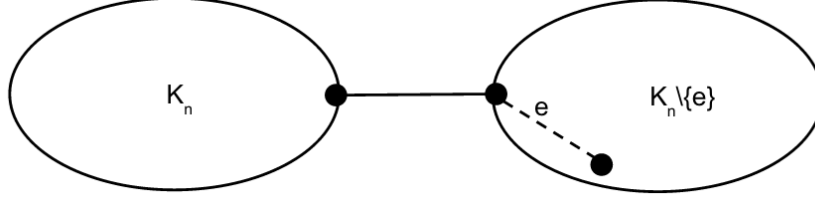


Figure 3: A graph G in which the only legal rewiring will disconnect the graph.

3.3 Degree sequences

It is clear from the definition of the algorithms above that the degree sequence in G will not be preserved under these rewirings. Indeed, by examining steps 4 and 5 in the algorithm statements, it can be seen that this algorithm always rewires edges from higher degree nodes to lower degree nodes. Although it might seem, based on this feature, that the graph would then tend toward regularity, it can be seen empirically that this is not the case. If we expect high degree nodes to be controlling large clusters, the edges that we are rewiring are those that are not actually involved in many triangles with that high degree node.

This being said, the algorithm can be modified to allow for a degree-preserving version. In this version, rather than rewiring one edge, we must rewire two edges at a time, as follows.

Degree Sequence Preserving Rewiring Algorithm 3

- 1: Find a pair $\{x, y\} \in E$ with a minimum value of $|N(x, y)|$
 - 2: Let $F \subset E$ be the set of edges in G that are both independent from $\{x, y\}$ and nonadjacent to $\{x, y\}$. Among these, choose an edge $\{u, v\}$ with the minimum value of $|N(u, v)|$.
 - 3: If $|N(u, x)| + |N(v, y)| > |N(u, y)| + |N(v, x)|$ and $|N(x, y)| + |N(u, v)| < |N(u, x)| + |N(v, y)|$, delete the edges $\{x, y\}$ and $\{u, v\}$, and replace them with the edges $\{x, u\}$ and $\{v, y\}$.
 - 4: Otherwise, if $|N(u, y)| + |N(v, x)| > |N(x, y)| + |N(u, v)|$, delete the edges $\{x, y\}$ and $\{u, v\}$, and replace them with the edges $\{x, v\}$ and $\{u, y\}$.
 - 5: If neither the conditions of steps 3 or 4 are met, return to step 2 and remove the edge $\{u, v\}$ from consideration.
 - 6: If no edge in F satisfies the requirements, return to step 1 and choose a different edge from G .
-

We note that this algorithm can be seen as a revision of Algorithm 2; a similar revision can be made for Algorithm 1. We further note that the benefit of degree sequence preservation here may be outweighed by the expense of such an algorithm; the computation time is substantially higher, since in step 2, we must consider substantially more edges in G than in the previous versions. As with Algorithms 1 and 2, it is straightforward to show that this algorithm is monotone with respect to the global clustering coefficient; indeed, the number of length-two paths here is constant, and hence the only change is that we are increasing the total number of triangles in G .

3.4 Complexity

Computationally, the most expensive step of both Algorithms 1 and 2 is the first, the calculation of the pair $\{x, y\} \in \bar{E}$ with the maximum value of $|N(x, y)|$, which corresponds to finding the maximum value of a subset of the elements of A^2 . In its most straightforward implementation, A^2 can be calculated in $\mathcal{O}(n^3)$ time, where n is the number of vertices in G . Isolating the appropriate subset

and finding its maximum can be done in $\mathcal{O}(n)$ time. Each iteration of the algorithm will change values of entries in A^2 . However, it will only change the values of entries in rows $A^2(x, :)$, $A^2(y, :)$, and $A^2(v, :)$ and columns $A^2(:, x)$, $A^2(:, y)$, and $A^2(:, v)$ in the case of Algorithm 1 ($A^2(u, :)$ and $A^2(:, u)$ are also affected in the case of Algorithm 2). These effects can be calculated *a priori* and the changes can be implemented through row/column updates which involve summing (or subtracting) a few appropriate rows of A and A^2 , which takes $\mathcal{O}(n)$ time. Thus, the full computation of A^2 only needs to be done once.

The rest of the algorithms consist of find the vertex $v \in N(x) \Delta N(y)$ with minimum f_v , checking that it meets the degree requirements, moving the edge (i.e. updating A), and updating A^2 . These can all be done in $\mathcal{O}(n)$ time for each iteration. Thus, the total cost of the algorithm is $\mathcal{O}(n^3 + kn)$, where n is the number of vertices in the graph and k is the number of edges to be rewired.

4 Simulated Results

We illustrated the effects of the rewiring algorithms by running them on simulated networks. We created Erdős-Rényi ($G_{n,p}$) networks, then iteratively rewired them using one of the rewiring procedures until there were no valid rewiring moves left. Unless otherwise stated, all networks were generated with 100 nodes and an edge density of 3%.

Both rewiring algorithms led to the global clustering coefficient of the network increasing monotonically. The “Find Best” algorithm (algorithm 1) had generally greater clustering gains than the “Improve Worst” algorithm (algorithm 2) (Figure 4). This is expected, as the “Find Best” algorithm is a global optimization procedure while the “Improve Worst” algorithm is a local optimization procedure.

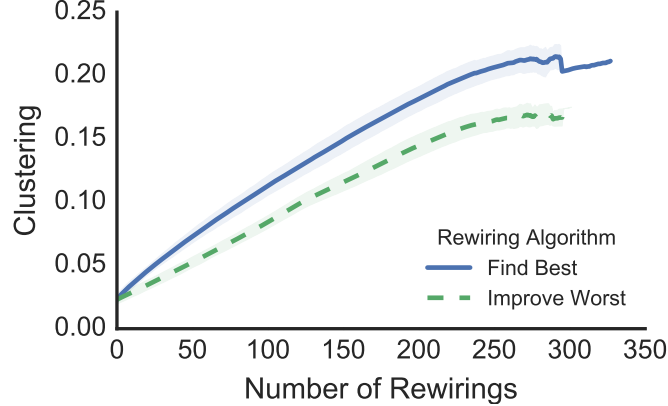


Figure 4: **Rewiring a network’s edges using either algorithm 1 or 2 raises its clustering coefficient.** The “Find Best” Rewiring Algorithm 1 (blue) optimizes the clustering coefficient increase globally rather than locally, leading to both a faster increase and a higher final clustering coefficient than the “Improve Worst” Rewiring Algorithm 2 (green). The solid lines show the mean increase in clustering coefficients over 100 networks and the shaded regions show one standard deviation.

For any network, the theoretical maximum number of rewires was the total number of edges in the network (i.e. rewiring every edge once). In practice, the rewiring algorithm terminated before all edges were rewired because eventually no unaltered edge could be moved in a way that improved clustering. However, the majority of the possible rewiring moves were still completed.

The simulated networks could be heavily rewired regardless of the number of nodes or edge density in each (Figure 5).

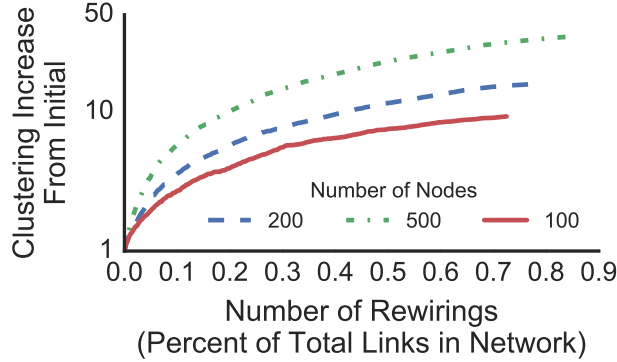


Figure 5: **Rewiring is possible for most edges in a network regardless of network size, but larger networks yield larger increases in clustering with rewiring.** The red, blue, and green lines represent Erdős-Rényi networks with $N = 100$, 200 , and 500 nodes, respectively. In each case, the network has an edge density of $1.5 \log(N)/N$. The plot shows the increase in clustering coefficient under “Find Best” Rewiring Algorithm 1, expressed as a multiple of the initial clustering coefficient before rewiring as the percentage of rewired edges increases. The larger the initial size of the network, the greater the impact of rewiring a set percentage of edges has on the increase in clustering coefficient.

Many rewires were still completed even when the rewiring algorithms were modified to preserve the degree distribution. While these procedures put greater requirements on what constituted a valid rewiring move, networks of all sizes and edge densities were able to have the majority of their edges rewired (Figure 6). However, the modified Rewiring Procedures did not produce as large of increases in clustering, as would be expected from the more limited options of where edges can be moved.

The rewiring algorithms are proven to increase a network’s global clustering coefficient, but there is another common measure of a network’s clustering: the average local clustering coefficient. The rewiring algorithms typically increase average local clustering coefficient, but they are not guaranteed to monotonically increase it. In fact, they do not (Figure 7). Both rewiring algorithms increase the local clustering over many rewires, but it is possible for the local clustering to stall or go down markedly during portions of the rewiring process.

In addition to clustering, the rewiring algorithms also altered other properties of the network structure. One such property is the path length. Path length increased during rewiring; as clusters formed it became harder to quickly navigate between them (Figure 8, left). However, the trade-off between clustering and path length was not constant. Clustering increased faster than path length during most of the rewiring, but in the end the path length increased faster. A network with a high clustering and low path length is commonly known as a small-world network [24], and the small-world index summarizes the relationship by stating the ratio of the clustering and the path length. A high small-world index indicates that a network has a particularly complex structure. The small-world index increased during the Rewiring Procedures as the clustering grew faster than the path length, but then plateaued and slightly decreased before the rewiring algorithms terminated (Figure 8, right).

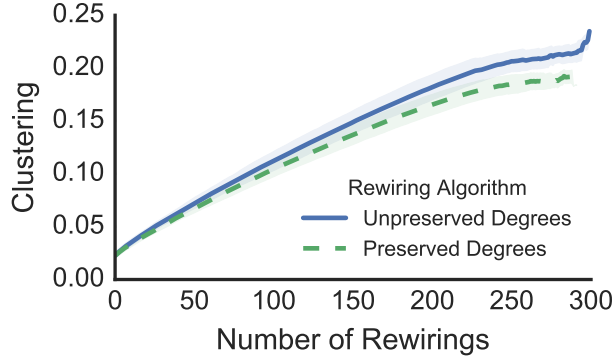


Figure 6: **Rewiring a network while preserving the degree sequence also increased clustering, though not as quickly.** Here, 100 instances of a 100 node Erdős-Rényi network were rewired using both the “Find Best” Rewiring Algorithm 1 (blue) and the Degree Preserving Rewiring Algorithm 3 (green). The solid lines show the mean increase and the shaded regions plot one standard deviation. Although both algorithms increased the clustering coefficient of the network, Rewiring Algorithm 1, which allowed for changes in the degree sequence of the network, did better than algorithm 3.

5 Concluding Remarks

We have introduced a new set of algorithms for rewiring edges in a network with the goal of maximally increasing the global clustering coefficient and minimally impacting other network properties, with a focus on preserving degree distribution and average degree length. Several variations of this algorithm were implemented, including one that preserves the degree sequence of the original network. We proved that the algorithms strictly increase the global clustering coefficient of a network, provided examples of local optima under Algorithm 1, and discussed the time complexity of running the algorithms.

Additionally, we ran the algorithms on a number of Erdős-Rényi ($G_{n,p}$) networks to examine how the clustering coefficient increased both as a function of the the number of nodes in the network (holding the edge density constant) and as a function of the number of edges rewired. We also examined how these were affected by the exact variation of the algorithm used. Other experiments examined how the increase in the average path length or the average local clustering coefficient compared to that of the global clustering coefficient. Our experiments corroborate the theoretical findings that the algorithms presented strictly increase the global clustering coefficient. They also show the average local clustering coefficient increases, though not necessarily monotonically.

Additionally, the simulation experiments demonstrate that the rewiring procedures create a small world. This is essentially the inverse of the process that Watts and Strogatz used when they introduced the small world concept [24]. They started with a regular lattice and rewired edges randomly until the network was fully random. Along the way the path length decreased faster than the clustering decreased, and so there was a period during the rewiring in which clustering was high and path length was low: a small world. Here, our rewiring procedures are doing the reverse by starting with a randomized network and deliberately rewiring edges to create a more regular network. And again, along the way a small world is created. Thus, the rewiring algorithms introduced here give a reciprocal story to how to build a small world network, given a fixed number of nodes and edges. These rewiring algorithms may be useful to those seeking to grow small world

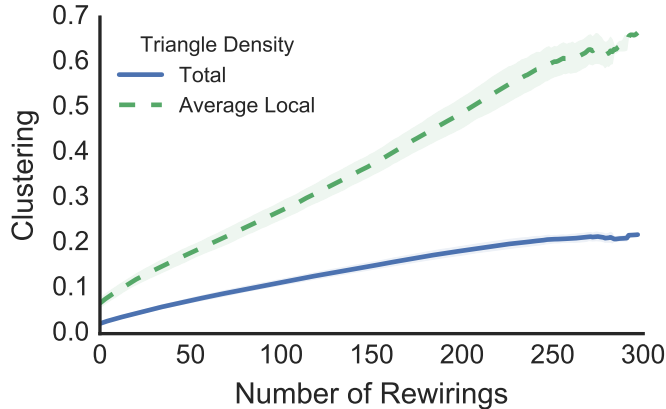


Figure 7: **Rewiring networks increased different kinds of clustering.** In Section 3, we proved that the rewiring algorithms increase a networks’ global clustering coefficient, which we show here averaged over 100 instances of a 100 node Erdős-Rényi network (blue). However, the average local clustering coefficient increases as well (green). The solid lines show the mean increase and the shaded areas show one standard deviation.

networks in engineered physical systems, or to explain how naturally-occurring physical systems construct themselves into small worlds.

Future work includes determining whether using approximations of the initial values of A^2 and $N(x, y)$ (the most computationally expensive element of the algorithm) affect the performance of the algorithm. One aspect of these algorithms which, for the sake of brevity, was not discussed in this paper is that of applications where these algorithms may be deployed. These include areas such as community detection and graph partitioning. Future work concerns research into these applications. Another important aspect of future work is determining, both experimentally and theoretically, the ideal number of edges which should be rewired for various applications in which modifying a network to increase its clustering coefficient is desirable.

Acknowledgments

J.A. was supported by the SUTD-MIT Postdoctoral Programme. The work of C.K. was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

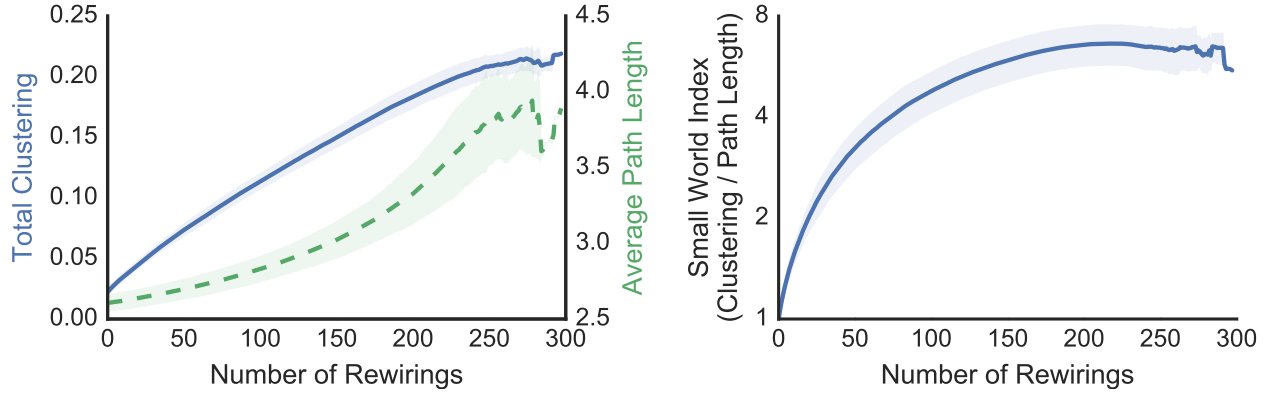


Figure 8: **The rewiring algorithm increases clustering faster than it increases path length, creating a small-world network.** Left: The change in both the global clustering coefficient (blue) and the average path length (green) are plotted for 100 instances of a 100 node Erdős-Rényi network. The solid lines show the mean change as a function of the number of edges rewired and the shaded areas plot one standard deviation. Initially, average path length increases along with the clustering coefficient. Right: We plot the change in the small world index, which is a ratio of the global clustering coefficient and the average path length, of the networks as a function of the edge rewirings. The rewiring process increases the small-world index by several multiples relative to the initial value.

References

- [1] F. ARRIGO AND M. BENZI, *Updating and downdating techniques for optimizing network communicability*, arXiv preprint arXiv:1410.5303, (2015).
- [2] A.-L. BARABÁSI AND R. ALBERT, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–512.
- [3] D. BARMPOUTIS AND R. M. MURRAY, *Networks with the smallest average distance and the largest average clustering*, arXiv preprint arXiv:1007.4031, (2010).
- [4] A. BARRAT AND M. WEIGT, *On the properties of small-world network models*, Eur. Phys. J. B, 13 (1999), pp. 547–560.
- [5] A. BEYGELZIMER, G. GRINSTEIN, R. LINSKER, AND I. RISH, *Improving network robustness by edge modification*, Physica A, 357 (2005), pp. 593–612.
- [6] S. BOCCALETTI, V. LATORA, Y. MORENO, M. CHAVEZ, AND D. HWANG, *Complex networks: structure and dynamics*, Phys. Rep., 424 (2006), pp. 175–308.
- [7] U. BRANDES AND T. ERLEBACH, eds., *Network Analysis: Methodological Foundations, Lecture Notes in Computer Science Vol. 3418*, Springer, New York, 2005.
- [8] A. CLAUSET AND C. MOORE, *How do networks become navigable?*, arXiv preprint arXiv:cond-mat/0309415, (2003).
- [9] A. CLAUSET, C. R. SHALIZI, AND M. E. J. NEWMAN, *Power-law distributions in empirical data*, SIAM Review, 51 (2009), pp. 661–703.

- [10] E. ESTRADA, *The Structure and Function of Complex Networks*, Oxford University Press, 2012.
- [11] P. HOLME AND B. J. KIM, *Growing scale-free networks with tunable clustering*, Phys. Rev. E, 65 (2002), p. 026107.
- [12] Z.-Y. JIANG, M.-G. LIANG, AND W.-J. AN, *Effects of efficient edge rewiring strategies on network transport efficiency*, Physica A, 394 (2014), pp. 379–385.
- [13] Z.-Y. JIANG, M.-G. LIANG, AND D.-C. GUO, *Improving network transport efficiency by edge rewiring*, Mod. Phys. Lett. B, 27 (2013), p. 1350056.
- [14] H. S. KOPPULA, K. PUSPESH, AND N. GANGULY, *Study and improvement of robustness of overlay networks*, tech. rep., Department of Computer Science & Engineering Indian Institute of Technology Kharagpur, January 2013.
- [15] A. LANCICHINETTI AND S. FORTUNATO, *Community detection algorithms: a comparative analysis*, Physical Review E, (2009), p. 056117.
- [16] D.-S. LEE, K.-I. GOH, B. KAHNG, AND D. KIM, *Evolution of scale-free random graphs: Potts model formulation*, Nuclear Physics B, 696 (2004), pp. 351–380.
- [17] W. LI-FU, W. QING-LI, K. ZHI, AND J. YUAN-WEI, *Enhancing synchronizability by rewiring networks*, Chin. Phys. B, 19 (2010), p. 080207.
- [18] V. H. P. LOUZADA, F. DAOLIO, H. J. HERRMANN, AND M. TOMASSINI, *Smart rewiring for network robustness*, Journal of Complex Networks, 1 (2013), pp. 150–159.
- [19] R. D. LUCE AND A. D. PERRY, *A method of matrix analysis of group structure*, Psychometrika, 14 (1949), pp. 95–116.
- [20] M. E. J. NEWMAN, *Random graphs with clustering*, Phys. Rev. Letters, 103 (2009), p. 058701.
- [21] J. I. PEROTTI, O. V. BILLONI, F. A. TAMARIT, D. R. CHIALVO, AND S. A. CANNAS, *Emergent self-organized complex network topology out of stability constraints*, Phys. Rev. Letters, 103 (2009), p. 108701.
- [22] F. RADICCHI, C. CASTELLANO, F. CECCONI, V. LORETO, AND D. PARISI, *Defining and identifying communities in networks*, Proc. Natl. Acad. Sci. U.S.A., 9 (2004), pp. 2658–2663.
- [23] A. SYDNEY, C. SCOGGIO, AND D. GRUENBACHER, *Optimizing algebraic connectivity by edge rewiring*, Applied Mathematics and Computation, 219 (2013), pp. 5465–5479.
- [24] D. J. WATTS AND S. H. STROGATZ, *Collective dynamics of ‘small-world’ networks*, Nature, 393 (1998), pp. 440–442.
- [25] D. R. WUELLNER, S. ROY, AND R. M. D’SOUZA, *Resilience and rewiring of the passenger airline networks in the united states*, Physical Review E, 82 (2010), p. 056101.
- [26] M. ZHOU AND J. LIU, *A memetic algorithm for enhancing the robustness of scale-free networks against malicious attacks*, Physica A: Statistical Mechanics and its Applications, 410 (2014), pp. 131–143.
- [27] T. ZHOU, G. YAN, AND B.-H. WANG, *Maximal planar networks with large clustering coefficient and power-law degree distribution*, Phys. Rev. E, 71 (2005), p. 046141.